267

DRAFT

ENCYCLOPEDIA OF THE HUMAN GENOME
2000

Hidden Markov Models

HMM, gene finding, profile HMM, training HMM, protein structure prediction

Przytycka, Teresa M
Teresa M Przytycka
Johns Hopkins School of Medicine USA

Hidden Markov Model is a statistical model frequently used for modelling biological sequences. In HMM, a sequence is modelled as an output of a discrete stochastic process, which progresses though a series of states that are "hidden" from the observer. At each state the process outputs a symbol from a finite alphabet (e.g. {A,T,C,G,ε}, where ε is an empty symbol) according to some probability distribution.

Introduction

A hidden Markov model (HMM) is a statistical model, initially developed for speech recognition (Rabiner, 1989), which has subsequently been used in numerous

biological sequence analysis applications. Current applications of HMMs in computational biology include among others modelling protein families (Eddy 2001, Krogh et al, 1993), gene finding, (Krogh et al. 1994, Burge et al, 1997, Lukashin et al, 1998, Henderson et al 1997, Salezberg, 1998), prediction transmembrane helices (2001) Krogh, , tertiary structure prediction (Bystroff, 2000; DiFrancesco et al 1997).

In a hidden Markov model, a biological sequence, e.g. a protein, DNA or RNA sequence, is modelled as an output generated by a stochastic process progressing through discrete time steps. At each time step, the process outputs a symbol (an amino acid or a nucleotide) and moves from one of a finite number of states to the next state. Frequently it is convenient to have states that do not output any symbol (e.g. a "delete" state in the profile HMM described below) thus, for uniformity, we include empty symbol, ε, in the alphabet. Both actions, the transition from state to state and the emission of a symbol, follow probability distributions, which are a part of the model. In a hidden Markov model, only the sequence of emitted symbols is observed. The path of states followed by the process is "hidden" from the observer.

Given a hidden Markov model, *M*, and a sequence *S*, the standard question is whether *S* has the property modelled by *M*. To address this question one needs to compute the probability, P[*S*|*M*], of sequence *S* being generated by *M*. The *log* of the ratio of P[*S*|*M*] to the probability of generating S by chance is usually used as a scoring function in assessing whether *S* has the model property. For example, given an HMM representing the globin protein family and a sequence of amino acids, the scoring function described above is used to determine if the given sequence belongs to the globins family.

Often hidden Markov models are designed in such a way that their states correspond to biologically relevant positions in the sequence. For example, in an HMM for gene finding, specific states may correspond to the beginning and to the end of an exon. In this case the most likely path of states used to generate the sequence contains information for annotating the sequence. Therefore, another important question concerning an HMM is finding the most likely path of states for a given sequence.

The following sections provide a formal definition of an HMM, describe efficient algorithms for finding the most likely path of states for a sequence, computing P[*S*|*M*], and finally describe methods of designing hidden Markov models.

Definitions

A first order hidden Markov model is defined formally as a tuple, M = (Q, Σ, *a*, *s*,, *e*) where:

- Q =.{1,…,*n*} is a finite set of states;
- $\Sigma = \{\sigma_1, \ldots, \sigma_m\}$ – is the alphabet, i.e. the set of output symbols;.
- *a* is *n* x *n* matrix of transition probabilities defined formally as $a(i,j) = \Pr[q_{t+1} = j | q_t = i]$, where q_t is the state visited at step *t*;
- *s* is *n* vector of start probabilities, i.e. $s(i) = \Pr[q_0 = i]$
- $e(i,j)$ is *n* x *m* matrix of emission probabilities defined formally as $e(i,j) = P[o_t = \sigma_j | q_t = i]$; where $o_t \in \Sigma$ is the symbol outputted in step *t*.

It is often convenient to have distinguished "start" and "end" states: *0*, and *n+1* that do not emit any symbols and remove vector *s* from the model definition. In the considerations below, we make this assumption. An HMM is usually visualized as a directed graph with vertices corresponding to the states and directed edges to the pairs of states *i, j* for which transition probability $a(i,j)$ is non-zero. A simple HMM is shown in Figure 1.

In a k^{th} order model the transition and emission probabilities depend on *k* last steps. Consequently, matrix *a* is of size n^{k+1} and matrix *e* is of size $n^k m$.

An HMM may generate the same sequence following different state paths (see Figure 1). Given an HMM M, sequence $S = o_1, \ldots, o_T$, and a path of states $p = q_0 q_1 \ldots q_m q_{T+1}$where $q_0=0$ and $q_{T+1} = n+1$, the probability of generating S using path *p* in model *M*, $P[S, p|M]$, is equal to the product:

$$P[S,p|M] = P[p|M]\ P[S|p,M]$$

where P[p|M] is the probability of the selecting the path p and P[S|p,M] the probability of generating sequence S assuming path p.

$$P[p|M] = a(q_0,q_1)\, a(q_1, q_2)\, a(q_2, q_3) \ldots a(q_T, q_{T+1});$$

$$P[S|p,M] = e(q_1,o_1)\, e(q_2, o_2) \ldots e(q_T, o_T)$$

The most likely path of a sequence S in model M is the path, p_{max}, that maximizes P[S,p| M] . Thus although the states of a hidden Markov are not directly observable, the most likely path provides information about the likely sequence of such "hidden" states.

Finally, the probability P[S|M] of generating sequence S, by an HMM, M is defined as:

$$P[S|M] = \Sigma_p P[S|M,p]$$

In practical applications probabilities values P are replaced with $-log$ P score. This avoids producing numbers that are too small to be represented by a computer.

Basic algorithms

Given an HMM, M, and a sequence, $S = o_1,\ldots o_T$ of length T, and assume that in step T+1 the process is in the stop state (n+1) generating empty symbol. The values p_{max} and P[S|M] can computed using a dynamic programming method. Let $v_k(i)$ be the most probable path that generates $o_1,\ldots o_i$ and ends at state k at step i. Obviously

$$p_{max} = v_{n+1}(T+1).$$

The recurrence for computing $v_k(i)$ is given by the following formula:

$$v_k(i) = e(k,o_i) \max_j v_j(i\text{-}1)a(j,k).$$

With appropriate initial conditions, the above recurrence provides the basis for an $O(n^2T)$- time dynamic programming algorithm known as the Viterbi algorithm. Since the number of states, n, is fixed for the model, the time of the algorithm depends linearly on the length of the input sequence.

Replacing maximum with summation in the recursive formula for $v_k(i)$ yields the recurrence for $[S|M]$. Namely, let $f_k(i)$ denote the probability of generating subsequence $o_1,\ldots o_i$ using a path that ends at state k. Then

$$f_k(i) = e(k,o_i)\Sigma_j f_j(i\text{-}1)a(j,k)$$

and

$$P[S|M] = f_{n+1}(T+1).$$

Variable $f_k(i)$, called the *forward variable*, is also used for computing the probability of state k in step i, $P[q_i = k|S,M]$. To compute the last probability, a similar *backward variable*, $b_k(i)$, is also used. Formally, $b_k(i)$ is the probability of generating the subsequence $o_{i+1},\ldots o_T$ using state k as the starting state and the usual "end" state $n+1$. The backward variable is computed similarly as the forward variable but the algorithm is executed in the "backward" direction: using "end" state in the place of the "begin" state. By definition of $f_k(i)$, $b_k(i)$ follows that:

$$P[q_i = k|S,M] = (f_k(i)\, b_k(i))/P[S|M].$$

Construction and training of an HMM

There are two basic steps in building of an HMM: designing the directed graph that describes the topology of the model (number of states, connections between states) and assigning transition and emission probabilities.

The topology of an HMM is usually designed in an ad-hoc way, based on the designer's understanding of the modelled sequence. Frequently, such a sequence can be described by a "grammar". For example, a simple grammar for a prokaryotic gene can be given as $S\ (C)^n\ E$ where S is the start codon, C is any codon different than an end codon and E is an end codon. C is repeated n times. In this case, it is natural to design the topology of an HMM in a way that follows the grammatical description. In the prokaryotic gene example, a topology implied by the simple grammar is shown on

Figure 2. The grammar, and subsequently a corresponding HMM, for the eukaryotic gene is far more complicated. It needs to describe a gene sequence as an interleaving sequence of exons and introns taking into account that the splicing can occur at any position a codon.

A different approach is taken in designing the so-called profile hidden Markov models for protein families. Namely, a universal topology is used and a correct setting of parameters elucidates the variations between families. The design includes "match" states, "insert" states and silent "delete" states (Figure 3).

In the second phase of the construction, the transition and emission probabilities are assigned to the model. This is done automatically based on a representative sample of sequences called the training set. The computational problem is described formally as follows:

Given a training set S_1, ..S_n and a topology of an HMM, M, find emission and transition probabilities that maximize the likelihood that S_1, ..S_n are generated by the model.

The usual assumption is that S_1, ..Sn are generated independently and therefore

$$P(S_1, ..S_n |M) = \Pi_i P(S_i |M)$$

And replacing the probability with -log score we have

$$\text{Score}(S_1, ..S_n |M) = \Sigma_i \text{Score}(S_i |M).$$

The training step is straightforward if for each training sequence, S_i, the paths of states, which the model uses to generate S_i, is known. In this case the training step reduces to collecting transition and emission frequencies along these paths. The training step becomes more sophisticated if the state paths are unknown. The main strategy in this case is to start with some initial probability distribution and the iteratively improve the model using the training set. For example, one frequently used method, the Expectation Maximization method approaches this problem as follows:

1. Assign some initial value to parameters (say uniform probability distribution).
2. For each sequence in the training set, compute the expected number of times each transition/emission is used. This can be done efficiently using the algorithms described in the previous section.

3. Estimate new parameters of the model based on the expected values from step 2.
4. Repeat steps 2 and 3 until some convergence criterion is reached.

It can be shown that Expectation Maximization method converges to a local maximum. Other training methods include the gradient descent method and simulated annealing**.**

One of the fundamental questions that one needs to consider during the training process is whether the training set contains enough data to estimate correctly transition and emission probabilities . Lack of data leads to overfitting of the model – the model cannot generalize the training data to a larger set. In particular, the question of sufficient data needs to be examined when deciding on the order of the model. In principle, a higher order model should be more accurate. For example, gene recognition models often are of 5^{th} order. (This is equivalent of keeping memory of two codons.) The number of parameters that need to be estimated grows exponentially with the order of the model and the possibility of overfitting increases.

References

Burge, C. and Karlin, S. (1997) Prediction of complete gene structures in human genomic DNA. *Journal of Molecular Biology* **268**: 78-94.

Bystroff C, Thorsson V, Baker D. (2000) HMMSTR: a hidden Markov model for local sequence-structure correlations in proteins. *Journal of Molecular Biology*. **301(1)** : 173-90.

Di Francesco F, Garnier J, and Munson PJ (1997) Protein topology recognition from secondary structure sequences: application of the hidden Markov models to the alpha class proteins. *Journal of Molecular Biology* **28 267(2)** : 446-63.

Eddy SR. (2001) HMMER: Profile hidden Markov models for biological sequence analysis (http://hmmer.wustl.edu/). 2001.

Henderson J, Salzberg S, Fasman KH. (1997) Finding genes in DNA with a Hidden Markov Model. *Journal of Computational Biology* **4(2)** :127-41.

Krogh A, Brown M., Mian IS, Sjolander K, and Haussler D.(1994) Hidden Markov models in computational biology: Application to protein modelling. *Journal of Molecular Biology*, **235** :1501-1531.

Krogh A, Larsson B, von Heijne G, Sonnhammer EL (2001) Predicting transmembrane protein topology with a hidden Markov model: application to complete genomes. *Journal of Molecular Biology* **305(3)** : 567-80.

Krogh A., Mian IS. and Haussler (1994) D. A hidden Markov model that finds genes in E. coli DNA. *Nucleic Acids Resarch* **22** : 4768-4778.

Lukashin AV. and Borodovsky M. (1998) GenMark.hmm: new solutions for gene finding. *Nucleic Acids Research* **26(4)** : 1107-1115.

Rabiner LR A tutorial on hidden Markov models and selected applications in speech recognition. *Proceeding of the IEEE* **77** : 257-286.

Bibliography

Clote P., Backofen R. (2000) *Computational Molecular Biology An Introduction*, John Wiley & Cons, LTD, Chichester UK.

Durbin R., Eddy S., Krogh A., Mitchison G. (1998) *Biological Sequence analysis*, Cambridge University Press, Cambridge UK.

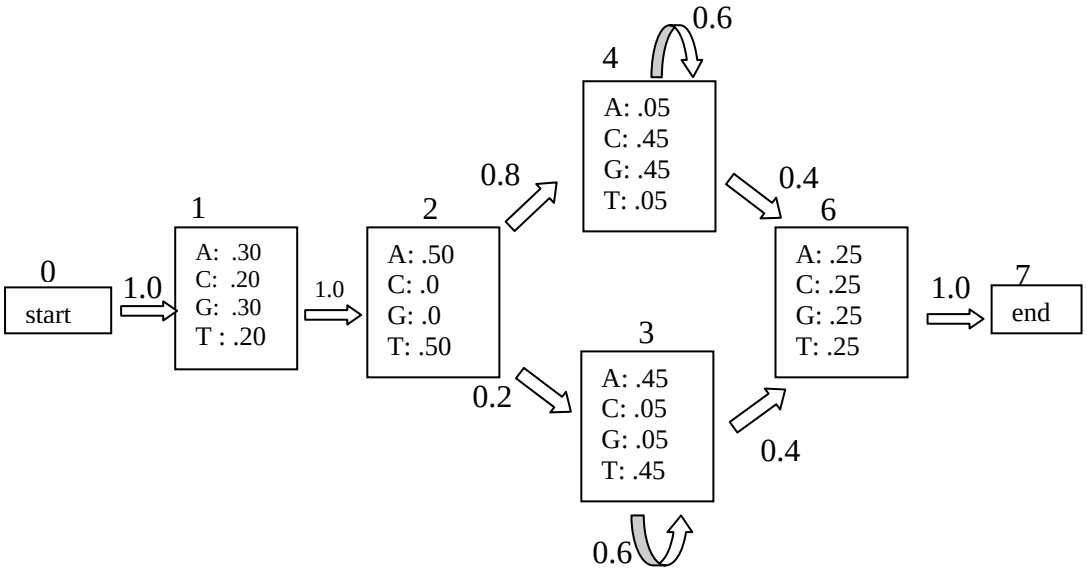

Figure 1. A simple hidden Markov model. The boxes correspond to states where the emmision probabilites for each state are given inside each box. The tranition probabilities are given above the corresponding arrows. Note that there are two state paths that can be used to generate sequence GAGCGCT: 0,1,2,4,4,4,4,6,7 and 0,1,2,3,3,3,3,6,7. The probability of generating the sequence using the first path is 1.06×10^{-4} and using the second path is 4.05×10^{-9}. The probability of generating the sequences by the model is the sum of these probabilities.

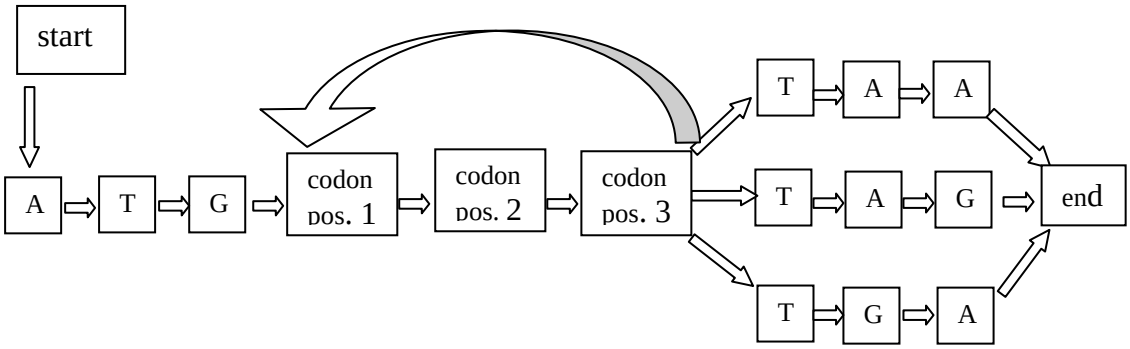

Figure 2. The topology of a simple HMM for prokaryotic gene recognition. In practice, the topology is more complex (e.g. Krogh 1994).

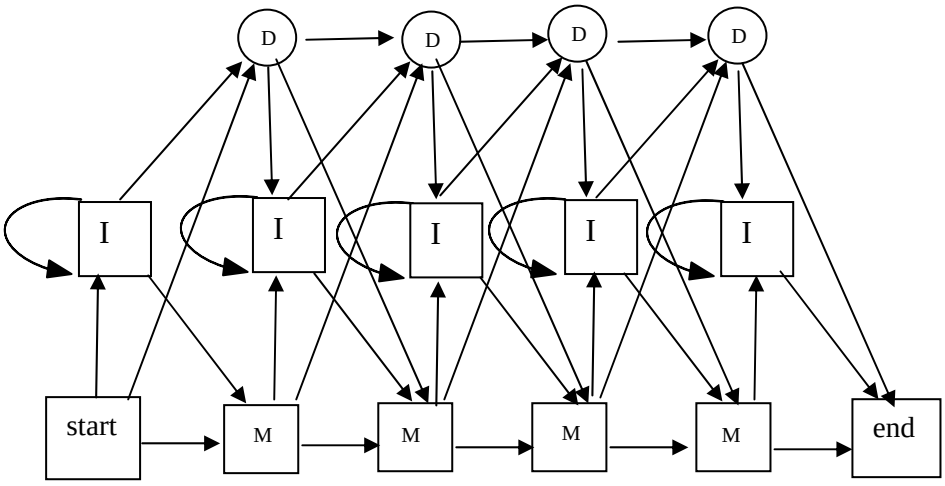

Figure 3. The topology of a profile HMM for a sequence family. The states labelled with M correspond to matches, the states labelled with I correspond to insertions and (silent) circle states correspond to deletions.